Concours Blanc : épreuve d'Informatique

PTSI Lycée Eiffel

3 juin 2019

Une après-midi au zoo.

Durée : 2H.

Ce sujet se propose d'aborder diverses situations pouvant mener à l'utilisation d'outils informatiques dans un parc zoologique.

Première partie : base de données des animaux du zoo.

Dans cette première partie, on souhaite construire une base de données contenant des informations importantes sur le fonctionnement du zoo. Cette base de données relationnelle contient notamment les tables suivantes :

- une table **Animaux**, contenant les attributs n°id (numéro d'identification de l'animal, qui sera la clé primaire de la table), espèce, nom, âge, date (correspond à la date d'arrivée de l'animal dans le zoo).
- une table **Employés**, contenant les attributs n°employe (numéro d'identification de l'employé servant également de clé primaire), nom, prénom, nbheures (nombre d'heures hebdomadaires effectuées par l'employé au zoo).
- une table **Tickets**, contenant un numéro de ticket nticket (qui sert de clé primaire), et des attributs date et tarif.

1. Écrire une commande SQL permettant de créer la table **Animaux**.
2. On souhaite ajouter une quatrième table **Soins** permettant de lister les soins effectués par les employés sur les animaux du zoo. Quelle clé primaire prendra-t-on pour cette table ? Elle devra contenir de plus des attributs naturedusoin et frequence (qui contient une chaine de caractères, « hebdomadaire » est par exemple une valeur possible de cet attribut). Citer les clés secondaires liées à la création de cette table.
3. Écrire une commande SQL permettant d'afficher le nom et l'âge de tous les éléphants du zoo.
4. Écrire une commande SQL permettant d'afficher le nombre de tickets « plein tarif » ayant été vendus le 12 mai 2018.
5. Écrire une commande SQL permettant d'afficher les noms et prénoms de tous les employés prodiguant des soins à l'ours nommé « Baloo ».

Deuxième partie : exploitation d'un fichier de données et programmation Python.

La liste des animaux du zoo est extraite de la base de données précédente pour former un fichier texte animaux.txt au format suivant : chaque ligne du fichier contient des données concernant un animal (et un seul), et plus précisément son espèce, son nom, son âge et sa date d'arrivée (au format jjmmaaaa) séparés par des ; (la caractère ; étant par ailleurs absent des données elle-mêmes). Ainsi, les deux premières lignes du fichier pourraient ressembler à ceci :

panda ;Huan Huan ;9 ;15012012
éléphant ;Dumbo ;78 ;01101941

1. Rappeler la syntaxe de la commande Python **open**, ainsi que les différentes options qu'on peut lui associer.
2. On rappelle que la commande Python **split(chaine,c)** permet de créer une liste à partir d'une chaîne de caractères en la séparant en morceaux, le caractère c servant de séparateur. Ainsi, split('panda ;Huan Huan ;9 ;15012012',' ;') va renvoyer la liste de chaines de caractères ['panda','Huan Huan','9','15012012']. Écrire une fonction Python **listealpha** qui renvoie la liste des noms (et uniquement des noms) de tous les animaux du zoo triée par ordre alphabétique, en allant chercher ces noms dans le fichier animaux.txt.
3. Écrire sur le même modèle une fonction Python qui renvoie la liste des noms des animaux du zoo étant agés de moins de 5 ans.

Troisième partie : étude de la température du logement des girafes (programmation Python).

Dans cette partie, on souhaite étudier des relevés de température effectués à l'intérieur du logement des girafes. On dispose pour celà de données sous la forme de deux listes Python de même taille, une liste **horaires** contenant les dates auxquelles ont été effectués les relevés de températures (on suppose que les données correspondantes sont des nombres entiers, correspondants à un nombre de minutes écoulées depuis une date de référence), et une liste **temperatures** contenant les températures mesurées à ces dates (sous formes de flottants, températures exprimées en degrés Celsius).

1. Rappeler comment les nombres flottants sont stockés dans les outils informatiques actuels.
2. Le capteur servant à effectuer les mesures de température stocke chacune des mesures effectuées sur 2 octets. Combien de chiffres significatifs peut-on espérer de telles mesures ?
3. La température du logement varie dans un intervalle dont l'amplitude ne dépasse pas les 30 degrés. On souhaite avoir une précision de l'ordre du centième de degré sur les mesures effectuées, le capteur choisi vous semble-t-il suffisant ?
4. En important le module habituellement utilisé pour ce genre de tracé (on écrira explicitement la commande sur la copie), écrire une suite d'instructions Python permettant de visualiser la courbe de température (en fonction du temps) issue des deux fichiers fournis.
5. Écrire une fonction Python **moyenne** prenant comme argument une liste de flottants L et renvoyant la moyenne des valeurs contenues dans la liste L.
6. Écrire une fonction Python **variance** prenant comme argument une liste de flottants L et renvoyant la variance des valeurs contenues dans la liste L. On rappelle que la variance d'une suite de réels $(x_1, \ldots, x_n)$ est le réel $V = \sum_{i=1}^{n}(x_i - \overline{x})^2 = \left(\sum_{i=1}^{n} x_i^2\right) - \overline{x}^2$, où $\overline{x}$ désigne la

moyenne de cette même suite de réels. Estimer la complexité de votre programme (nombre d'opération élémentaires effectuées en fonction du nombre n de réels dans la liste).

7. On souhaite que la température moyenne du logement des girafes soit comprise dans l'intervalle $[25, 28]$ et que sa variance ne dépasse pas 3. Écrire une fonction Python **checktemp** prenant comme argument une liste de flottants (supposée être une liste de mesures de température), et vérifiant si elle satisfait à ces conditions (on pourra bien sûr reprendre les programmes des deux questions précédentes, même si on n'est pas certain d'en avoir écrit des versions correctes).
8. On considère qu'il y a eu une anomalie dans le relevé de températures quand on y repère trois valeurs successives en-dessous de 22. Écrire une fonction Python **detectionanomalies** qui prend comme arguments les deux listes horaires et temperatures, et qui renvoie une liste des dates auxquelles on a repéré une anomalie (la date retenue sera celle de la troisième mesure consécutive en-dessous de 22 degrés).

Quatrième partie : étude de la courbe de croissance d'un bébé panda (analyse numérique).

Le zoo a eu le bonheur de voir naitre un bébé panda il y a deux semaines ! La courbe de croissance du bébé pendant ses six premiers mois est censée suivre une équation du type $P'(t) = \alpha P(t) + \beta P^2(t)$, où P représente le poids en kilogrammes du bébé et α, β sont deux paramètres réels constants. On donne de plus la condition initiale $P(0) = 0.1$.

1. Quel type de courbe obtiendrait-on si le paramètre β était nul ? En déduire le signe probable des constantes α et β si on souhaite que le modèle soit crédible.
2. Rappeler le principe de la méthode d'Euler pour la résolution des équations différentielles du premier ordre.
3. Écrire une fonction Python **simulpanda(a,b)** prenant comme arguments deux paramètres réels a et b (ce sont les constantes α et β de l'énoncé) et effectuant le tracé de la solution de l'équation sur l'intervalle $[0, 10]$ en appliquant la méthode d'Euler.
4. On dispose d'une liste Python **poids** contenant les poids effectivement mesurés pour le bébé panda lors de ses 10 premiers jours (une mesure par jour), qu'on souhaite comparer aux valeurs données par le modèle théorique pour des valeurs fixées des constantes a et b, obtenues à l'aide du programme de la question précédente. Le modèle (et les valeurs des constantes) seront considérées comme correctes si l'écart relatif maximal entre les mesures effectives et les valeurs théoriques ne dépasse pas 5%. Écrire un programme Python **valide(a,b,poids)** qui effectue cette vérification et renvoie un booléen comme résultat (True ou False). On fera bien attention au fait que le programme **simulpanda** écrit à la question précédente renverra (probablement) beaucoup plus que 10 valeurs si on souhaite que la méthode d'Euler soit efficace.
5. Pour déterminer les valeurs adéquates des constantes α et β, on peut aussi effectuer une interpolation polynômiale à partir des valeurs obtenues dans la liste **poids**. Mais pour celà il faut d'abord déduire de ces valeurs de $P(t)$ des valeurs (approchées) de $P'(t)$. Si on note Δt le temps (supposé constant) entre deux mesures successives du poids, une façon d'obtenir ces données est d'utiliser l'approximation $P'(t) \simeq \dfrac{P(t + \Delta t) - P(t - \Delta t}{2\Delta t}$.
   (a) En utilisant cette approximation, écrire une fonction Python **approxder(poids,delta)** qui prend comme arguments une série de mesures du poids du panda ert l'intervalle de temps delta entre deux mesures successives, et qui renvoie une liste de valeurs approchées de P' (qui ne fera pas exactement la même longueur que la liste poids !).

(b) La documentation de la fonction **polyfit** du module numpy précise les choses suivantes :

- la fonction polyfit(x,y,deg) cherche les coefficients à donner à un polynôme P de degré deg pour que la relation $y_i = P(x_i)$ soit la meilleure possible (où x_i et y_i sont les éléments de même indice des listes x et y).
- les paramètres x et y sont deux listes de valeurs de même taille, deg est un entier.

Écrire une commande Python permettant d'obtenir les valeurs optimales de α et de β à partir des données **poids** et **approxder(poids,delta)**, utilisant la fonction polyfit décrite ci-dessus.